Amendments to the claims (this listing replaces all prior versions):

1. (previously presented) A method comprising

maintaining a database persistently on a physical storage medium,

partitioning the database into regions based on characteristics of items of data in the database,

receiving tasks that may be in contention to write data to the respective regions,

reducing the contention among the received tasks by creating jobs that are based on each of the tasks, each of the jobs needing to write data in no more than one of the regions,

for each of the regions, queuing only those jobs that need to write data in the region and executing the queued jobs for each region, one job after another and without contention, and

executing jobs that are queued for different regions simultaneously, in parallel, and without contention.

2. (previously presented) The method of claim 1 in which the items of data of the database comprise objects in an object database.

3. (previously presented) The method of claim 1 in which the items of data are provided as objects to an object-oriented application.

4. (original) The method of claim 1 in which an object relational broker provides persistent storage of objects for an object-oriented application.

5. (previously presented) The method of claim 1 in which the database comprises a relational database with object-oriented extensions.

6. (previously presented) The method of claim 1 in which the database comprises files that persistently store the items of data.

7. (previously presented) The method of claim 1, 2 or 3 in which the number of tasks received is arbitrarily large.

8. (previously presented) The method of claim 1, 2, or 3 in which the tasks are received from task sources, and the number of task sources is arbitrarily large.

9. (previously presented) The method of claim 1, 2, or 3 also including assigning each of the regions to a respective available processor, each of the regions being assignable to any of the processors.

10. (previously presented) The method of claim 9 in which the number of the regions is no fewer than the number of available processors.

11. (previously presented) The method of claim 9 in which the number of the regions is equal to the number of available processors.

12. (previously presented) The method of claim 9 also including organizing the regions to maximize the throughput of the available processors in executing the jobs.

13. (previously presented) The method of claim 9 in which the regions are assigned dynamically to the available processors to maximize the throughput of the available processors.

14. (previously presented) The method of claim 1, 2, or 3 in which the tasks are received asynchronously.

15. (previously presented) The method of claim 1, 2, or 3 in which the tasks are received concurrently.

16. (previously presented) The method of claim 9 in which the processors do not use shared memory.

17. (previously presented) The method of claim 1, 2, or 3 in which creating the jobs comprises creating a hierarchy of subtasks in which the lowest level of the hierarchy contains the jobs.

18. (original) The method of claim 1, 2, or 3 in which at least one of the tasks comprises a single job.

19. (previously presented) The method of claim 1, 2, or 3 in which one of the jobs generates a task to be performed.

20. (previously presented) The method of claim 1, 2, or 3 in which each of the tasks is completed with a certainty that is at least as high as the certainty that items of data updated in a requested database transaction are not lost once the transaction is committed.

21. (previously presented) The method of claim 1, 2, or 3 in which at least one of the regions comprises a single data item.

22. (previously presented) The method of claim 1, 2, or 3 in which at least one of the regions comprises at least a million data items

23. (canceled).

24. (previously presented) The method of claim 9 in which more than one of the regions is associated with one of the available processors.

25. (original) The method of claim 24 in which each of the processors comprises a physical processor running at least one process.

26. (original) The method of claim 1, 2, or 3 in which each of the tasks is generated by a user request.

27. (previously presented) The method of claim 9 in which each of the regions is associated with at least two available processors, one of which executes jobs and the other of which performs administrative functions with respect to the associated region.

28. (previously presented) The method of claim 1, 2, or 3 in which queuing the jobs includes maintaining a queuing system that has a capacity to receive jobs for execution at an arbitrarily large rate by adding processors proportionately to the number of jobs expected to require execution.

29. (original) The method of claim 28 in which the queuing system includes conceptual rows each of which can receive the jobs.

30. (original) The method of claim 29 in which each of the rows is locked when jobs are being received in the row.

31. (original) The method of claim 30 in which a job can be accepted by the corresponding processor for execution from any of the rows that is not locked.

32. (previously presented) The method of claim 9 in which each region comprises millions of sub-regions.

33. (original) The method of claim 1, 2, or 3 in which additional jobs are created in connection with the execution of the jobs.

34. (original) The method of claim 33 in which further jobs are created by the additional jobs.

35. (original) The method of claim 33 in which the creation of the additional jobs is dependent on data read from the database in executing the jobs.

36. (previously presented) The method of claim 9 in which additional jobs are created in connection with the execution of the jobs, and a process running on one of the processors executes the jobs and creates the additional jobs, and in which at least some of the additional jobs are queued for regions served by other processors.

37. (original) The method of claim 1, 2, or 3 in which the tasks relate to commercial transactions.

38. (previously presented) The method of claim 9 in which each of the jobs is assigned an index associated with the corresponding region.

39. (previously presented) The method of claim 38 in which the indexes are used to load balance the jobs among the processors.

40. (original) The method of claim 1, 2, or 3 in which the database includes database units that are distributed among different physical locations.

41. (original) The method of claim 1, 2, or 3 in which each of the jobs comprises steps.

42. (original) The method of claim 41 in which execution of a job includes executing a portion of the steps, committing a database transaction representing those steps, and repeating until the job is completed.

43. (original) The method of claim 42 also including, upon a failure to complete any portion of the steps, restarting the execution at the first step of the failed portion with at least the same level of certainty that the job will be completed as the certainty that data written in a requested transaction is not lost once the transaction is committed.

44. (original) The method of claim 31 in which a row is locked only for reading when a processor is accepting jobs from that row.

45. (previously presented) The method of claim 9 in which (1) queuing the jobs includes maintaining a queuing system that has a capacity to receive jobs for execution at any arbitrarily large rate, (2) the queuing system includes conceptual rows each of which can receive the jobs,

and (3) the queuing system includes conceptual columns associated with respective contention spaces, the queuing system comprising a conceptual matrix of cells at the intersections of the rows and columns, in which each of the cells may be read from or written to without conflicting with reads and writes to other cells.

46. (original) The method of claim 45 in which the rows are associated with sources of jobs and the number of rows is sufficient to permit all of the sources of jobs to load jobs into the queue concurrently.

47. (original) The method of claim 45 in which the number of rows is sufficient to permit jobs to be fetched for execution from all of the columns concurrently.

48. (original) The method of claim 1, 2, or 3 also including synchronizing the executions of synchronization groups of the jobs to ensure correctness of results.

49. (original) The method of claim 48 in which the synchronizing includes assigning to each of the jobs of a synchronization group a tag that identifies them as members of the group.

50. (original) The method of claim 48 in which the synchronizing includes assigning to each of the jobs of a synchronization group a quorum fraction representing the job's proportion of participation in the group.

51. (original) The method of claim 50 in which the jobs are not executed until all of the jobs in the synchronization group are ready for execution by a processor.

52. (previously presented) Apparatus comprising

a physical storage medium on which a database is persistently maintained, the database being partitioned into regions, and

a job processing mechanism that (1) receives tasks that may be in contention to write data to the respective regions of the database, (2) creates jobs that are based on each of the tasks, each of the jobs needing to write data in no more than one of the regions, (3) for each of the regions, queues only those jobs that need to write data in the region, one job after another and without contention, and (4) executes jobs that are queued for different regions simultaneously, in parallel, and without contention.

53. (previously presented) The apparatus of claim 52 in which the data items of the database comprise objects in an object database.

54. (previously presented) The apparatus of claim 52 in which the data items are provided as objects to an object-oriented application.

55. (currently amended) The ~~method~~ apparatus of claim 52 in which an object relational broker provides persistent storage of objects for an object-oriented application.

56. (currently amended) The ~~method~~ apparatus of claim 52 in which the data is stored in a relational database with object-oriented extensions.

57-83. (Canceled).